
PERGAMON Computers and Mathematics with Applications 39 (2000) 117–121

An International Journal
computers & mathematics with applications

www.elsevier.nl/locate/camwa

The Optimal On-Line Parallel Machine Scheduling

YONG HE
Department of Applied Mathematics, Zhejiang University
Hangzhou 310027, P.R. China
heyong@math.zju.edu.cn

(Received July 1999; accepted August 1999)

Abstract—This paper investigates on-line parallel machine scheduling problems. We show the optimality of the classical LS algorithm.

Keywords—Analysis of algorithm, On-line scheduling, Worst-case ratio.

1. INTRODUCTION

In the parallel identical machine scheduling problem, we are given a set $\mathcal{J} = \{p_1, p_2, \ldots, p_n\}$ of independent jobs, each with a positive processing time, that must be scheduled on m parallel and identical machines. We identify the jobs with their processing times. The jobs and machines are available at time zero, and no preemption is allowed. The objective is to minimize the maximum machine completion time (makespan) $C_{\max}$, or to maximize the minimum completion time $C_{\min}$. These NP-complete problems [1], usually denoted by $P||C_{\max}$ and $P|\mathbb{C}_{\min}$, have many applications in practice [2,3]. A scheduling problem is called *on-line* if it requires the scheduling of jobs irrevocably on the machines as soon as they are given, without any knowledge about jobs that follow later on. If we have full information on the job data before constructing a schedule, this problem is called *off-line.*

In a worst-case analysis, the performance of an on-line algorithm is measured through the worst-case ratio with respect to the optimal solution of the off-line problem. For a set $\mathcal{J}$ of jobs and an approximation algorithm A, let $C_A(\mathcal{J})$ denote the minimum machine completion time produced by the algorithm A and let $C^*(\mathcal{J})$ denote the minimum machine completion time produced by an optimal algorithm in an off-line version. Let $w_A(\mathcal{J})$ denote the makespan produced by the algorithm A and let $w^*(\mathcal{J})$ denote the optimal makespan in an off-line version. Then the worst-case ratio of the algorithm A is defined as

$$R_A = \inf_{\mathcal{J}} \left\{ \frac{C_A(\mathcal{J})}{C^*(\mathcal{J})} \right\}, \qquad \text{for problem } P||C_{\min},$$

$$R_A = \sup_{\mathcal{J}} \left\{ \frac{w_A(\mathcal{J})}{w^*(\mathcal{J})} \right\}, \qquad \text{for problem } P||C_{\max}.$$

This research is supported by National Fundamental Research Project of China and National Natural Science Foundation of China (Grant Number 19701028).

0898-1221/00/$ - see front matter Typeset by AMS-TEX
PII: S0898-1221(00)00070-5

The simplest algorithm for the on-line parallel machine scheduling problem is the *list scheduling* (LS for short) algorithm which was introduced by Graham [2]. This algorithm always assigns the current job to the machine with minimum workload on it at the moment. Applying LS to $P\|C_{\max}$, Graham showed $R_{\mathrm{LS}} = 2 - 1/m$. Faigle, Kern and Turán [4] further observed that it is the best possible on-line algorithm for two and three machines. It means that there is no deterministic on-line algorithm for $P\|C_{\max}$ with a worst-case ratio better than $3/2$ and $5/3$ for $m = 2, 3$, respectively. For $m \geq 4$, several algorithms have been proposed which have a slightly better worst-case ratio than the LS algorithm in [5–8]. Recently, He and Zhang [9] proved that if the processing times of all jobs are in the interval $[p, rp]$, where $p > 0$ and $r \geq 1$, then the worst-case ratio of LS is tightened to $(r+1)/2$ for $m = 2$ and $r < 2$. They also show that LS is still the best possible on-line algorithm for any value of r. In this paper, we will show that for this kind of tightly-grouped processing times, this property still holds for each $m \geq 3$ when r is small.

Regarding the problem $P\|C_{\min}$, the off-line version is deeply studied. Deuermeyer, Friesen and Langston [3] proved the worst-case ratio of the *longest processing time* (LPT for short) algorithm is at least $3/4$. Csirik, Kellerer and Woeginger [10] showed the exact worst-case ratio of this algorithm is $(3m-1)/(4m-2)$. Woeginger [11] presented a polynomial time approximation scheme for this strong NP-hard problem. But to the author's knowledge, not much literature considers the on-line version of this problem. In this note, we will consider this on-line problem. We still assume that the processing times of all jobs are in the interval $[p, rp]$, where $p > 0$ and $r \geq 1$. By presenting the worst-case ratio of the LS algorithm, we conclude that it is the best possible on-line algorithm for each m and r.

This paper is organized as follows. Section 2 deals with the on-line problem of $P\|C_{\max}$ and Section 3 investigates the on-line problem of $P\|C_{\min}$.

2. MINIMIZING THE MAKESPAN

This section assumes that the jobs arrive one by one and the processing times of all jobs are in the interval $[p, rp]$, where $p > 0$, $r \geq 1$. We consider the on-line problem $P\|C_{\max}$. As mentioned above, LS is not the best possible algorithm when $m > 3$, but this section will imply that, if the processing times of all jobs are well tightly-grouped, LS is still the best. Theorem 2.1 is cited from [9].

THEOREM 2.1. *Applying LS to the problem $P\|C_{\max}$, if all jobs have their processing time within interval $[p, rp]$, where $p > 0$, $r \leq m/(m-1)$, then*

$$\frac{w_{\mathrm{LS}}(\mathcal{J})}{w^*(\mathcal{J})} \leq 1 + \frac{(m-1)(r-1)}{m}. \qquad \blacksquare$$

THEOREM 2.2. *Any on-line algorithm A has a worst-case ratio*

$$R_A \geq 1 + \frac{(m-1)(r-1)}{m},$$

if all jobs have their processing time within interval $[p, rp]$, where $p > 0$, $r \leq m/(m-1)$.

PROOF. Consider the following instances. The first m jobs $p_1, \ldots, p_m$ have the same processing time 1. If an algorithm A assigns at least two of them to the same machine, then no further job comes any more. It follows $w_A/w^* = 2 > 1 + (m-1)(r-1)/m$. If they are assigned to different machines by the algorithm A, the next m jobs $p_{m+1}, \ldots, p_{2m}$ with processing time r come. If A does not assign them to different machines, then no new job comes, and hence, $w_A/w^* = (2r+1)/(r+1) > 1 + (m-1)(r-1)/m$. Otherwise, jobs $p_{2m+1} = \cdots = p_{3m} = r$ come again. Continue the above arguments until the jobs $p_{m(m-2)+1} = \cdots = p_{m(m-1)} = r$ come. If the above last m jobs are not assigned to different machines, then $w_A/w^* = ((m-1)r +$

$1)/((m-2)r+1) > 1+(m-1)(r-1)/m$. If A assigns them to different machines, then the next and last job $p_{m(m-1)+1} = r$ comes. We thus have $w_A/w^* = 1+(m-1)(r-1)/m$. ∎

From Theorem 2.1 and Theorem 2.2, we conclude that LS is the best possible on-line algorithm for each m when $r \le m/(m-1)$.

3. MAXIMIZING THE MINIMUM MACHINE COMPLETION TIME

This section still assumes that the jobs arrive one by one and the processing times of all jobs are in the interval $[p, rp]$, where $p > 0$, $r \ge 1$. We consider the on-line problem $P\|C_{\min}$.

THEOREM 3.1. *Applying the LS algorithm to the above problem, the worst-case ratios are*

(i) $R_{\mathrm{LS}} \ge 1/m$, *for* $r \ge m$,
(ii) $R_{\mathrm{LS}} \ge 1/r$, *for* $1 \le r < m$.

PROOF. We will prove these two ratios by contradiction separately. By normalizing all jobs, we can assume that $p = 1$.

(i) Denote by $l(M_i)$ the workload of the machine M_i after assigning all jobs in the LS schedule, $i = 1, \ldots, m$. Without loss of generality, we can suppose the $l(M_1) \ge l(M_2) \ge \cdots \ge l(M_m) = C_{\mathrm{LS}}(\mathcal{J})$. Let p_t be the last job assigned to M_1 in the LS schedule, and s is the starting time of p_t. Then

$$l(M_1) = s + p_t. \tag{1}$$

Suppose that the result does not hold, that is to say,

$$C_{\mathrm{LS}}(\mathcal{J}) < \frac{C^*(\mathcal{J})}{m}. \tag{2}$$

By average argument, we have

$$(m-1)l(M_1) + C_{\mathrm{LS}}(\mathcal{J}) \ge \sum_{i=1}^{m} l(M_i) = \sum_{i=1}^{n} p_i \tag{3}$$

and

$$\sum_{i=1}^{n} p_i \ge mC^*(\mathcal{J}). \tag{4}$$

Combining (2)–(4),

$$l(M_1) \ge \frac{m+1}{m} C^*(\mathcal{J}). \tag{5}$$

Since p_t is not assigned to M_m in the LS schedule, we get

$$s \le l(M_m) < \frac{C^*(\mathcal{J})}{m}. \tag{6}$$

From (1), (5), and (6), we know $p_t \ge C^*(\mathcal{J})$. It implies that, in an optimal schedule, the machine processing job p_t does not process any other job. Hence,

$$C^*(\mathcal{J}) \le \frac{1}{m-1}\left(\sum_{i=1}^{n} p_i - p_t\right). \tag{7}$$

If $p_t > C^*(\mathcal{J})$, we are allowed to decrease the processing time of job p_t to $C^*(\mathcal{J})$ because of the following two reasons.

(a) Obviously, it does not change the optimal solution value.
(b) Since p_t is the last job of the machine with maximum workload in the LS schedule, this operation does not change the LS schedule and $C_{\mathrm{LS}}(\mathcal{J})$ does not increase.

Hence, we can assume that $p_t = C^*(\mathcal{J})$. By (1), (3), and $s \leq C_{\mathrm{LS}}(\mathcal{J})$, we have

$$(m-1)p_t + mC_{\mathrm{LS}}(\mathcal{J}) \geq \sum_{i=1}^{n} p_i. \tag{8}$$

Substituting $p_t = C^*(\mathcal{J})$ and (7) into (8), we have

$$\begin{aligned} mC_{\mathrm{LS}}(\mathcal{J}) &\geq \sum_{i=1}^{n} p_i - (m-1)p_t = \left(\sum_{i=1}^{n} p_i - p_t\right) - (m-2)p_t \\ &\geq (m-1)C^*(\mathcal{J}) - (m-2)C^*(\mathcal{J}) = C^*(\mathcal{J}). \end{aligned} \tag{9}$$

This is the desired contradiction.

(ii) In the same way as above, we suppose that $C_{\mathrm{LS}}(\mathcal{J}) < C^*(\mathcal{J})/r$. Since the processing time of any job is at least 1, we can assume that $C_{\mathrm{LS}}(\mathcal{J}) \geq 1$. (Otherwise, the LS algorithm yields the optimal schedule.) Hence, we have $C^*(\mathcal{J}) \geq r$. This inequality implies that, in the optimal schedule, every machine processes at least two jobs. Let k be the number of jobs on the machine which determines $C^*(\mathcal{J})$. We know $|\mathcal{J}| \geq 2m$ and $C^*(\mathcal{J}) \leq kr$. In order to get a contradiction, we will prove $C_{\mathrm{LS}}(\mathcal{J}) \geq k$.

Obviously, the workload of the machine containing two jobs is at least 2. In the LS schedule, if there is a machine M_i which processes only one job, then there must be another machine M_j processing at least three jobs since $|\mathcal{J}| \geq 2m$. According to the LS rule, the workload of M_i is not less than the starting time of the third job of M_j, and hence, is at least 2. Therefore, we conclude that every machine of the LS schedule has workload at least 2. It means that $C_{\mathrm{LS}}(\mathcal{J}) \geq 2$. Combining it with our assumption, we currently have that $C^*(\mathcal{J}) \geq rC_{\mathrm{LS}}(\mathcal{J}) \geq 2r$. Hence, we know that $|\mathcal{J}| \geq 3m$. Repeating the above arguments, we can deduce that $C_{\mathrm{LS}}(\mathcal{J}) \geq 3$, ..., $C_{\mathrm{LS}}(\mathcal{J}) \geq k$. This desired result finishes the proof. ∎

THEOREM 3.2. *Any on-line algorithm has a worst-case ratio as follows:*

$$R_A \leq \frac{1}{m}, \qquad \text{for } r \geq m \qquad \text{and} \qquad R_A \leq \frac{1}{r}, \qquad \text{for } r < m.$$

PROOF. For $r \geq m$, consider the following instance. The first m jobs $p_1, \ldots, p_m$ both have a processing time of 1. If an algorithm A assigns them to different machines, the next and last $m-1$ jobs $p_{m+1}, \ldots, p_{2m-1}$ with a processing time m comes, the makespan $C_A = 1$ while the optimum makespan $C^* = m$. It follows that $C_A/C^* = 1/m$. If at least two of the first m jobs are assigned to the same machine by the algorithm A, then no further job comes any more.

It follows $C_A/C^* = 0$. Therefore, we conclude that for any on-line algorithm A, $R_A \leq 1/m$.

For $r < m$, the instance is almost the same except that the processing time of the jobs $p_{m+1}, \ldots, p_{2m-1}$ is r if they are needed. ∎

From Theorem 3.1 and Theorem 3.2, we conclude that LS is the best possible on-line algorithm for any m and r.

REFERENCES

1. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, (1979).
2. R.L. Graham, Bounds for certain multiprocessing anomalies, *Bell System Tech.* **45**, 1563–158 (1966).
3. B.L. Deuermeryer, D.K. Friesen and M.A. Langston, Scheduling to maximize the minimum processor finish time in a multiprocessor system, *SIAM J. Algorithms Discrete Methods* **3**, 190–196 (1982).
4. U. Faigle, W. Kern and G. Turán, On the performance of on-line algorithm for particular problems, *Acta Cybernetica* **9**, 107–119 (1989).
5. S. Albers, Better bounds for on-line scheduling, In *Proc. 29th Annual ACM Symp. on Theory of Computing*, pp. 130–139, (1997).

6. Y. Bartal, A. Fiat, A. Karloff and R. Vohra, New algorithm for an ancient scheduling problem, In *Proc. 24^{th} Annual ACM Symp. on Theory of Computing*, pp. 51–58, (1992).
7. G. Galambos and G. Woeginger, An on-line scheduling heuristic with better worst-case ratio than Graham's list scheduling, *SIAM J. on Computing* **22**, 349–355 (1993).
8. D.A. Karger, S.J. Phillips and E. Torng, A better algorithm for an ancient scheduling algorithm, *J. of Algorithms* **20**, 400–430 (1996).
9. Y. He and G. Zhang, Semi on-line scheduling on two identical machines, *Computing* **62**, 179–187 (1999).
10. J. Csirik, H. Kellerer and G. Woeginger, The exact LPT-bound for maximizing the minimum completion time, *Oper. Res. Letters* **11**, 281–287 (1992).
11. G. Woeginger, A polynomial time approximation scheme for maximizing the minimum machine completion time, *Oper. Res. Letters* **20**, 149–154 (1997).
12. H. Kellerer, Bounds for nonpreemptive scheduling jobs with similar processing times on multiprocessor systems using LPT-algorithm, *Computing* **46**, 183–191 (1991).